NASA Technical Memorandum 83550

A Real-Time, Portable, Microcomputer-Based Jet Engine Simulator

R. A. Blech, J. F. Soeder,
and J. R. Mihaloew
Lewis Research Center
Cleveland, Ohio

Prepared for the
1984 Simulators Mini-Conference
sponsored by the Society for Computer Simulation
Norfolk, Virginia, April 18-20, 1984

NASA

A REAL-TIME, PORTABLE, MICROCOMPUTER-BASED JET ENGINE SIMULATOR

R. A. Blech, J. F. Soeder, and J. R. Mihaloew
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

Modern piloted flight simulators require detailed models of many aircraft components, such as the airframe, propulsion system, flight deck controls and instrumentation, as well as motion drive and visual display systems. The amount of computing power necessary to implement these systems can exceed that offered by dedicated mainframe computers. One approach to this problem is through the use of distributed computing, where parts of the simulation are assigned to computing subsystems, such as microcomputers. One such subsystem, a real-time, portable, microcomputer-based jet engine simulator, is described in this paper. The simulator will be used at the NASA Ames Vertical Motion Simulator facility to perform calculations previously done on the facility's mainframe computer. The mainframe will continue to do all other system calculations and will interface to the engine simulator through analog I/O. The engine simulator hardware includes a 16-bit microcomputer and a floating-point coprocessor. There is an 8 channel analog input board and an 8 channel analog output board. A model of a small turboshaft engine/control is coded in floating-point FORTRAN. The FORTRAN code and a data monitoring program run under the control of an assembly language real-time executive. The monitoring program allows the user to display and/or modify simulator variables on-line through a data terminal. A dual disk drive system is used for mass storage of programs and data. The CP/M-86 operating system provides file management and overall system control. The frame time for the simulator is 30 milliseconds, which includes all analog I/O operations.

INTRODUCTION

The accuracy and detail of a simulation is determined by the requirements of the simulation task, the complexity of the physical system being simulated, and the capabilities of the simulation computer. Piloted flight simulation is perhaps one of the most demanding simulation tasks. A complex aircraft system must be represented mathematically, and that representation must execute on a computer in real-time. The computer must also control various flight deck instrumentation, motion drive and visual display systems.

In order to accomplish goals such as the evaluation of propulsion system and airframe control interactions, representative airframe and propulsion system models are necessary. However, the detail of these models is often constrained by the storage and speed limitations of the simulation computer. Extensive simplification of some of the aircraft subsystem models is then required to allow real-time operation. This process not only compromises the fidelity of the model, but can be costly and time consuming as well. Of course, a faster and more powerful simulation computer could be acquired, but this is not always practical.

One approach to solving this problem is to distribute the computational load among many "tightly coupled" processors that communicate with each other through a specialized network (1). This method is currently being investigated, but is not yet mature. It does, however, offer the most potential for increasing computational throughput with current computer technology. Another, simpler approach is to offload part of the computations to a separate computer system (this system, if necessary, can itself consist of several "tightly coupled" computers) which is "loosely coupled" to the simulation mainframe. The communication between this separate computer system and the mainframe could be digital (serial or parallel), or through standard A/D and D/A converters. In either case, the communication overhead must be much less than that of the simulation computations if any benefit is to be realized.

The latter approach is the one being taken by NASA Lewis Research Center in a distributed processing experiment to be conducted at the NASA Ames Research Center's Vertical Motion Facility (VMS). The VMS (2) is a piloted simulator which, in addition to a six-degree-of-freedom type cockpit, is also capable of ±30-ft of vertical motion. This feature is particularly useful for rotorcraft and V/STOL applications. Normally, all simulator calculations for the VMS are done on the facility's mainframe computer. However, for the distributed processing experiment, the calculations for the airframe, flight deck instrumentation, motion and visual display systems will be performed on the mainframe, while the propulsion system calculations will be performed on a separate microcomputer system. The microcomputer hardware and software, which form a stand-alone, portable, real-time simulator, are the subject of this paper.

ENGINE SIMULATOR REQUIREMENTS

The system to be modeled for this simulation experiment was a small turboshaft engine. A typical turboshaft engine is shown in figure 1. The turboshaft propulsion system consists of a compressor, combustor, gas generator turbine and a power turbine. The engine control consists of a hydromechanical unit (HMU), that provides basic engine control and fuel metering, and an electrical control unit (ECU), that trims the HMU to maintain rotor speed and load sharing (in the 2-engine helicopter application). The ECU also limits the gas generator turbine outlet temperature. A more complete description of the engine and control system can be found in reference 4.

At NASA Ames Research Center, the engine simulator must interface to the VMS mainframe which runs the helicoper airframe simulation in a 30 millisecond frame time. Selection of the same frame time for the engine simulator provides a 3 Hertz bandwidth, which was deemed sufficient for the piloted simulator application. Using the same frame time also simplifies the interfacing of the engine simulator to the mainframe. This I/O interface must be relatively simple for low software overhead, but of sufficient speed to provide the required data throughput. It is also desirable to have some type of operator interface to allow display and/or modification of simulation variables. This capability should reside on the simulator and consume a small portion of the frame time. It is highly desirable to have the engine simulation code written in a high level language and to have sufficient hardware and software support for the simulator's microcomputer to permit downloading of the code from a program development computer (mainframe) to the simulator. Programming of the engine simulation in a high level language (i.e., FORTRAN) necessitates the use of real-time programming techniques (3).

The microcomputer hardware itself must meet the speed requirements of the particular application. It is also desirable that the hardware be compact and easily transported. The technology should be of sufficient maturity such that adequate hardware and software support is available. This includes such items as a standard bus, I/O and peripheral hardware, and high level language support.

HARDWARE DESCRIPTION

The simulator hardware is shown in figure 2. It consists of a microcomputer board, a disk drive controller board, and analog I/O boards. The boards are housed in a 9 slot card cage which has a power supply and cooling fans. Presently, 4 of the 9 slots are used, leaving 5 slots available for future expansion. All user interaction is accomplished through the attached CRT and keyboard. Two floppy disk drives are used for program and data storage. A printer is available for hard copies of simulator data.

The microcomputer board is based on the Intel 8086 microprocessor and the 8087 floating-point coprocessor. The board has 64K of RAM, parallel and serial I/O, programmable timers and an interrupt controller (5). The fast floating-point capability provided by the 8087 is a major reason for its selection. Operation of the 8087 is mostly transparent to the user, especially if a high level language is used. A FORTRAN compiler and an assembler which support both the 8086 and 8087 are available. The microcomputer board as well as the other system boards are interconnected via Intel's Multibus (5) system. Use of the Multibus ensures a wide variety of peripheral and other hardware support.

The analog boards provide 8 channels of analog to digital conversion (ADC) and 8 channels of digital to analog conversion (DAC). These analog channels are used to communicate with the mainframe computer. Although noise immunity and accuracy suffer somewhat with this approach (as opposed to digital interfacing) these disadvantages are outweighed by the greatly reduced software and hardware overhead. Additionally, since many computer systems support analog I/O operations, the simulator is not limited to one specific hardware and software environment, such as the NASA Ames VMS mainframe computer.

SOFTWARE DESCRIPTION

Engine and Control Model

The engine model was derived by applying basic aerothermodynamic principles for each component. Steady state component characteristics were used to define the energy and mass transfer across each boundary. Low frequency dynamics, associated with rotor inertias and component heat soaks were retained. High frequency volume dynamics were omitted since these were considered to be outside the frequency bandwidth required in piloted simulations. Algebraic loops that resulted from omitting high frequency dynamics were made to converge by using a high gain digital integrator technique. The resulting dynamic effect is similar to that resulting from application of the conservation equations. Computing times were reduced and component performance map accuracy was retained by using a curve fitting technique similar to regression analysis. The maps were curve fit using segmented polynomial functions.

Since control evaluation is a prime consideration in the NASA Ames application, a detailed control representation was essential. Accuracy of the control model was assured by deriving it directly from the control specification diagrams used in the real control.

The resulting analytical model is a sixteenth order model which includes two real engine states, three psuedo engine states and eleven control states. The resulting set of differential equations was solved as a general initial and boundary value problem using a trapezoidal integration algorithm. The function generation and convergence techniques were designed specifically to accomplish rapid computation and stability.

The engine model was coded in floating-point FORTRAN. The availability of a FORTRAN compiler for the 8086-8077 microcomputer greatly simplified the transporting of the code from the program development mainframe to the microcomputer. Only minor changes to the original mainframe code were necessary to achieve successful compilation on the microcomputer's compiler. These changes typically involved removal of I/O statements that were no longer necessary for operation on the microcomputer, which had its own library of I/O routines.

There are several inputs to the model which, in this case, are generated externally on the VMS mainframe computer (figure 3). These inputs are routed by the mainframe from either the airframe calculations or the VMS cockpit mockup. The first input represents a load torque to the engine model. It is generated by the airframe simulation and is a result of the aerodynamic forces on the rotor blade. Another input, called the load demand spindle (LDS), is a function of the rotor blade collective pitch. The collective pitch is controlled by the helicoper pilot. The power available spindle (PAS) input sets the upper limit on available engine power, and is also set by the pilot. Finally, a power turbine reference speed signal, which sets the desired rotor speed, is fed to the engine control simulation. Environmental conditions are determined from the external inputs of altitude and mach number. In

response to these inputs, the engine simulation outputs a generated torque value, as well as outputs for cockpit displays such as rotor speed and gas turbine inlet temperature. These inputs and outputs are transferred via the analog interface of the microcomputer-based simulator hardware.

Assembly Language Executive

The engine simulator is an interrupt-driven system. An interrupt is generated internally every 30 milliseconds to begin the simulator frame. This interrupt is generated by the microcomputer's on-board programmable timer. The analog input board used an interrupt to signal the microcomputer that converted data is available. The 8087 also signals any floating-point error conditions via interrupt. Finally, a failsafe timer generates an interrupt if any external device fails to respond to a data read or write request within a specified interval.

An assembly language executive was developed to efficiently handle the response to these interrupts. The executive also initializes the simulator system before each run. During the simulation frame, after all analog data has been read, the FORTRAN engine model is called by the executive. After the model calculations are completed, the analog output channels are updated. The remaining time in the frame is spent executing the data monitoring routines. These routines continue to execute until the next timer interrupt is received to begin the next simulator frame.

Assembly language routines perform the physical analog input and output operations through the DAC and ADC hardware. These routines also define the logical connection between the hardware DAC or ADC channel and a corresponding FORTRAN simulation variable. The logical connection consists of an address, which specifies the FORTRAN variable associated with the analog channel, and a scale factor. The scale factor is used in the conversion of the FORTRAN floating-point representation from/to the integer form used by the DAC and ADC hardware. Both the address and the scale factor may be modified on-line, thus allowing any DAC or ADC channel to be reassigned. This feature is particularly useful for dynamic debugging of the simulator in a stand-alone mode.

Data Monitoring Software

The Microcomputer Interactive Data System or MINDS is a software package that can be used for on-line data display and parameter modification. The MINDS software is loaded into the microcomputer memory, co-resident with the simulation code and the real-time executive. Between the time that the processor has finished computing the simulation and the time that it gets the next timer interrupt, the real-time executive can call the MINDS software to allow operator interaction with the simulation. This interaction takes the form of the operator assigning a name and data type to certain memory locations. This process of assigning attributes to memory locations defines MINDS data elements. The operator can then display and modify the data elements symbolically while the simulation is running.

In addition, tables of the data elements can be defined thereby allowing display of a group of data elements simultaneously. The MINDS package contains commands to manipulate the data elements including the capability to save data element definitions in a disk file so that they can be used at a later time. Finally, the MINDS package contains a monitor to display and set any group of locations in the microcomputer memory referenced by absolute address. The monitor also has the ability to set conditional breakpoints based on data element values anywhere in the executing code. Once the breakpoint has been reached, the monitor will collect the values of all the microcomputer registers and, optionally, a predefined data table. This capability facilitated the debugging of the real-time simulation.

CP/M Operating System

The CP/M operating system is a general purpose, single user operating system marketed by Digital Research, Inc. of Pacific Grove, Ca. The operating system provides the facilities to do console communications, program load and unload, disk file management and rudimentary memory management. The operating system contains three major sections, the Command Control Processor (CCP), the Basic Disk Operating System (BDOS), and the Basic Input/Output System (BIOS). The CCP accepts all commands that are typed from the console, interprets them, and takes appropriate action. The BDOS contains the facilities to read, write and organize files on a floppy disk. Finally, the BIOS contains all the hardware-dependent information necessary to allow CP/M to operate in a particular environment.

In the engine simulator application, the CP/M-86 operating system is used to load the simulation, executive and MINDS software from the floppy disks. In addition, the MINDS software uses the operating system for disk file management of data element definitions. Further information on CP/M-86 can be found in reference 7.

SOFTWARE DEVELOPMENT CYCLE

The software development cycle is shown in figure 4. The development began with the engine model formulation and FORTRAN coding on an IBM 370 mainframe at NASA Lewis Research Center. Model functionality and accuracy were verified on the IBM 370. At this point, the source code was downloaded via a serial link from the IBM 370 to a microcomputer development system. The source code was recompiled on the development system's resident FORTRAN compiler. The compiler generated native code for the 8086-8087. The resulting object code was linked with the real-time executive and the MINDS object code. The final module was then transferred to a CP/M-86 compatible disk for conversion to an executable command file on the engine simulator hardware.

Timing studies on the hardware verified that the 30 ms frame time requirement was met. Next, steady state data were collected and compared to data obtained from the IBM 370 version of the FORTRAN simulation. After the steady state data had been verified, transient data were collected. The inputs to the simulator, which would typically be provided by the VMS mainframe, were simulated internally. The main control input, LDS, was stepped and various outputs were observed through the DAC channels and a strip chart recorder. The step response of the microcomputer-based simulator was compared to that of the IBM 370 simulation and verified.

Once operation of the simulator was verified at NASA Lewis on a stand-alone basis, steps were initiated to transport the unit to the VMS facility at

NASA Ames. There, it will be interfaced to the mainframe computer via the analog I/O. Testing will again be performed, including "open-loop" tests where the simulator reads data from the mainframe (which executes a resident engine simulation), but the mainframe does not receive input data from the simulator. The analog outputs will be monitored, however, to verify that correct results are being generated. The final stage of testing will be complete when system operation is verified with the VMS being completely dependent on the microcomputer-based engine simulator for all engine calculations.

SIMULATOR OPERATION

The simulator is initialized by bootstrapping the CP/M-86 operating system. The simulator code resides on a disk as a CP/M ".CMD" file which identifies it as an executable command file. It is invoked by typing the name which precedes the ".CMD" suffix. Thus the file "ENGSIM.CMD" is invoked by typing "ENGSIM". The simulator code is then loaded into system memory by CP/M-86. The system starts up initially in the MINDS data monitoring software. This allows any data element definition files to be loaded. After this step, MINDS is exited by the "." command. The interrupts are then enabled and the engine simulation code is executed. The simulator is initially in the "IC" or initial condition mode. The simulator mode is selected locally through the keyboard or remotely by the mainframe via an analog channel. Any of the following modes are available: "IC" or initial condition, "RUN" and "HOLD". These modes operate similarly to an analog computer. At any time, simulation variables may be displayed and/or modified since the MINDS software is continually operating during the frame idle time.

Severe error conditions such as exponent overflow, zero divide and analog output overflow are detected by the 8087. The 8087 sends an interrupt to the 8086 to notify it of the error condition. The 8086 responds by changing the simulator mode to "HOLD" and displaying a message on the CRT. The message contains information as to the type of error and the location of the error. The operator can then attempt to recover from the error and continue running or abort the simulation.

PERFORMANCE

Timing

Simulator performance on a stand-alone basis has been verified. The simulator timing diagram is shown in figure 5. The overall simulation frame time is 30 ms, although the actual computation time for the FORTRAN engine simulation is 21 ms. The analog I/O operations take a total of 1.4 ms. The remaining 7.6 ms is used for operator I/O through MINDS. Experience has shown that actually only 4 to 5 ms is required for adequate response by the MINDS software. This means that at least another 2.6 ms is available for additional FORTRAN model computations, if needed. Thus for a 30 ms frame time on this simulator hardware configuration, 23.6 ms is actually available for the FORTRAN engine model calculations, and 6.4 ms is required for I/O overhead.

Accuracy

Steady state data obtained from the microcomputer-based simulator compares almost exactly with that obtained from the IBM 370 FORTRAN simulation. Worst case differences were less than .01 percent. Since the IBM 370 and the 8087 coprocessor handle such things as rounding, floating-point format and function evaluation (i.e., square root, etc.) slightly differently, discrepancies of the type observed were expected.

It should be noted that the 8087 implements the proposed IEEE floating-point arithmetic standard. The IEEE standard is intended to eliminate most of the inconsistencies in floating-point arithmetic among various computer systems. By adhering to this standard, discrepancies in floating-point calculations among different computers can be minimized.

The steady state results were also compared to those obtained from a more detailed engine model, programmed in the continuous system modelling program (CSMP) on the IBM 370. The CSMP model matched data from the engine manufacturer's steady state cycle deck. The CSMP results and the microcomputer-based simulator results agreed typically to within 1 to 3 percent. These differences were attributed to the polynomial curve-fit approach used for the component maps on the microcomputer-based simulator. The CSMP simulation uses table look-up and interpolation methods.

The CSMP model of the engine system is more detailed dynamically, and hence has greater fidelity than the model used for the microcomputer-based simulator. Transient data from the simulation adequately matched those from the CSMP model within the frequency range of interest for the piloted simulator application.

CONCLUDING REMARKS

A real-time, portable, microcomputer-based jet engine simulator has been described. The hardware and software specifications have been presented. A helicopter engine and control model, coded in floating-point FORTRAN, has been run on the simulator in a 30 ms frame time. The accuracy and frequency content of the simulation are of the level required for piloted simulations. The simulator has been evaluated on a stand-alone basis. The engine simulator will be integrated into the NASA Ames VMS piloted simulator facility where its performance will be evaluated.

The hardware described in this paper is only one approach to constructing a microcomputer-based simulator. It is anticipated that advances in the hardware area will make higher performance, lower cost systems possible. The availability of fast, floating-point hardware has been shown to simplify the software development cycle considerably. Real-time simulations that were once only achievable through the use of integer, assembly language techniques are now possible using high level languages and floating-point arithmetic.

REFERENCES

(1) Blech, R. A. and Arpasi, D. J., "An Approach to Real-Time Simulation Using Parallel Processing," NASA TM-81731, 1981.

(2) Wilcox, D. E. and Quigley, H. C., "V/STOL Aircraft Simulation - Requirements and Capabilities at Ames Research Center," AIAA Paper 78-1515, 1978.

(3) Mihaloew, J. R., "A Nonlinear Propulsion System Simulation Technique for Piloted Simulators," NASA TM-82600, 1981.
(4) "T700 Training Guide," SEI-418, General Electric Co., Aircraft Engine Group, Lynn, MA., 1977.
(5) Barthmaier, Joe, "Intel Multibus Interfacing," iSBC Application Handbook, Intel Corporation, Santa Clara, 1981, pp. 1-45 to 1-78.
(6) "iSBC 86/12A Single Board Computer Hardware Reference Manual", manual 9803074-01, Intel Corporation, 1979.
(7) Zaks, Rodnay, The CP/M Handbook with MP/M, Sybex, Berkeley, 1980, 321 pp.

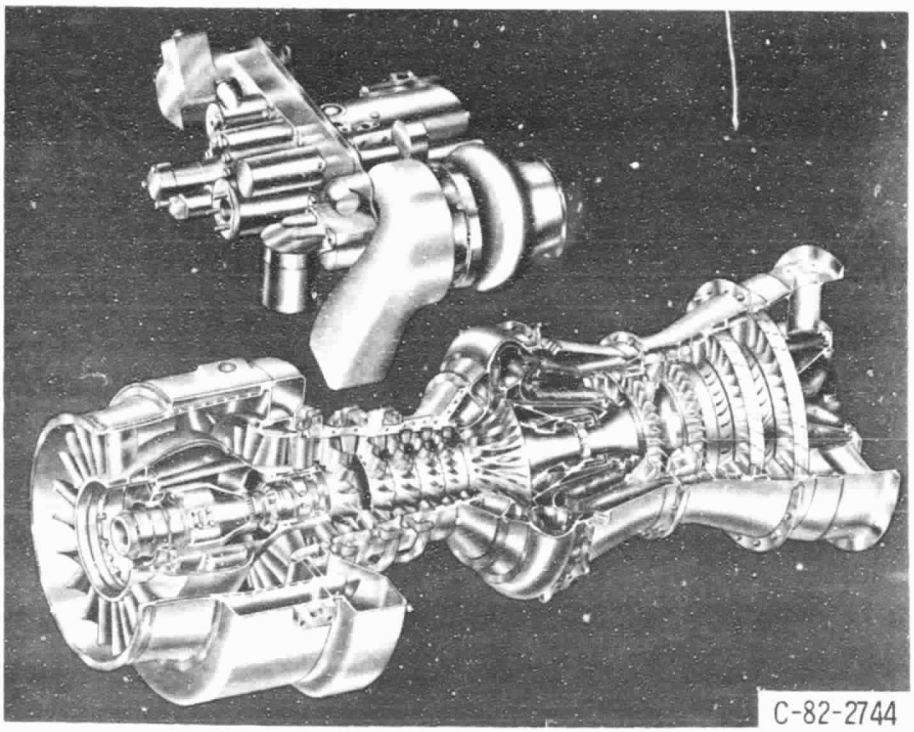

Figure 1. - Small turboshaft engine.

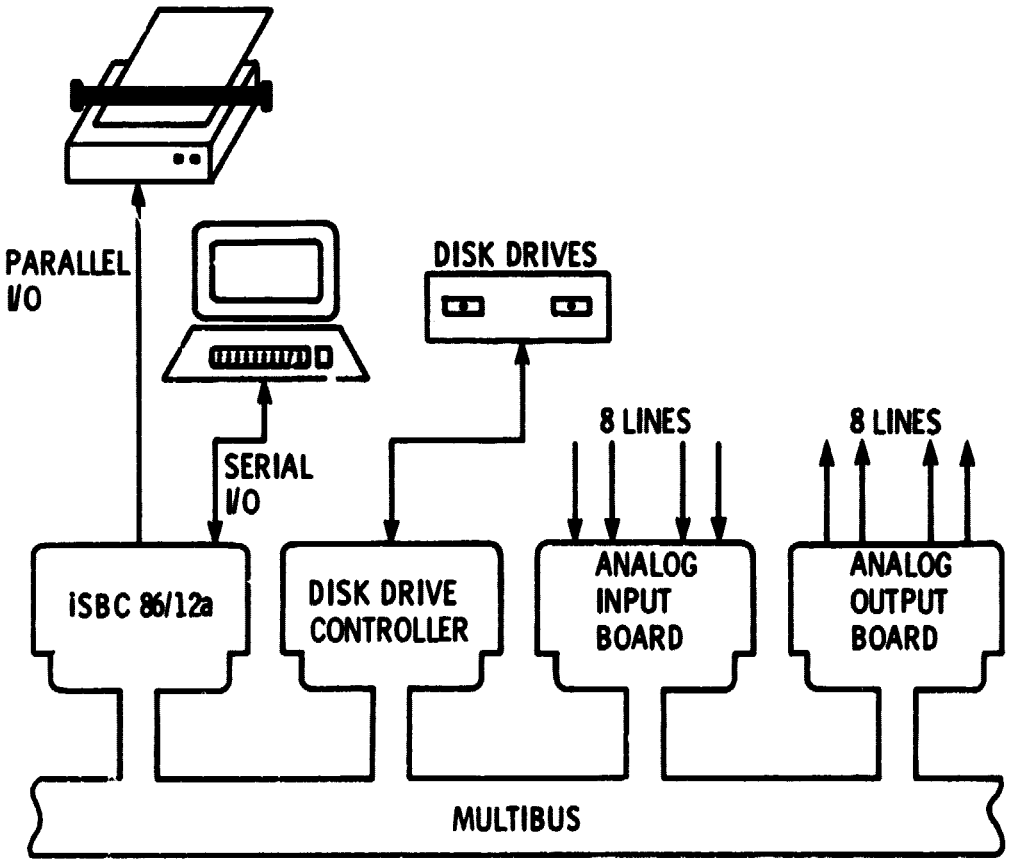

Figure 2. - Simulator hardware configuration.

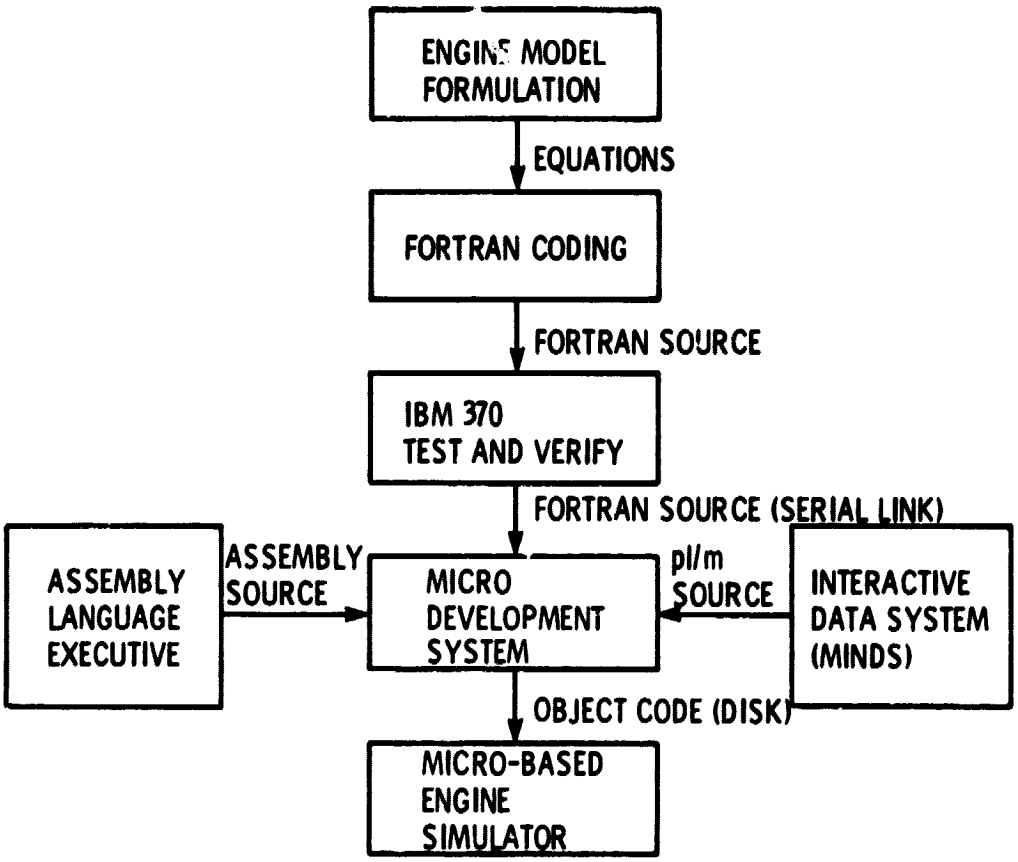

Figure 3. - Software development cycle.

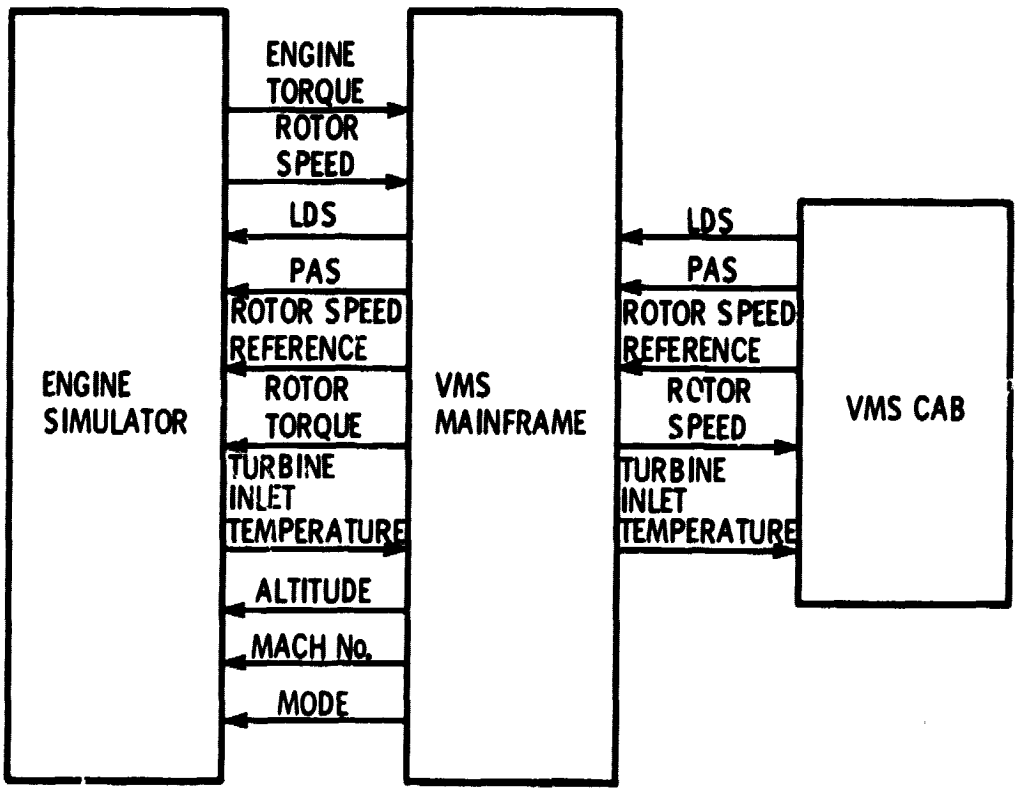

Figure 4. - Engine simulator-VMS interface.

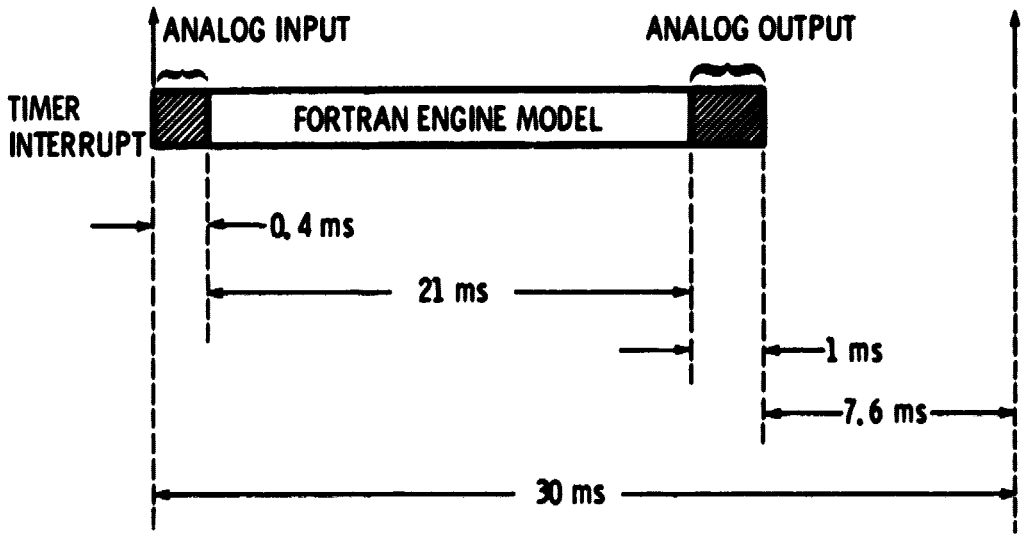

Figure 5. - Engine simulator timing.